Creating an Infrastructure for Ubiquitous Awareness

Michael Kantor & David Redmiles

Information & Computer Science

University of California, Irvine 92697

{mkantor, redmiles}@ics.uci.edu

Abstract: Much research has examined the use of awareness tools for enhancing coordination. However, the proposed tools tend to either make people aware of one information source in great detail or a variety of sources with very little detail. We refer to this problem as the detail-variety trade-off. Another problem with awareness tools is that they are not interchangeable. Specifically, users of these tools have only limited ability to route awareness information to different types of awareness tools that might better suit their work styles. We believe these two problems are primarily software engineering problems to which we present a solution called CASS (Cross Application Subscription Services), for obtaining awareness information from diverse sources and routing it to a variety of awareness tools.

Keywords: Event and Notification Server Architectures, Awareness, Push Technologies

1 Introduction

Awareness is a well recognized technique for enhancing coordination among people and between people and software (Dourish, 1992, Gutwin, 1998). Our primary focus is on creating awareness in order to enhance the efficiency with which people work together. However, techniques presented in this paper are fully applicable to communicating awareness information between software and people, providing a new and highly flexible technique by which background processes, including agents, can communicate with users. While the primary focus of this paper is a software engineering framework for providing awareness, it is primarily of interest to HCI researchers and practitioners for whom awareness is frequently a key technique. Awareness is used to enhance usability by providing users with feedback about the state of a system (Gaver, 1991), and peripheral information related to a task (Williamson and Shneiderman, 1992) (Sibert and Jacob, 2000). This paper addresses two key flaws in standard awareness tools; the detail-variety trade-off and the lack of interchangeability of awareness styles.

Detail-Variety Trade-off

A key problem with current work in awareness tools is what we refer to as the detail-variety trade-off. An awareness tool must have an information source from which it obtains awareness information. Some awareness tools access very few information sources, but provide users with detailed descriptions or illustrations of recent events from those sources. This creates a strong awareness of activity but only within these few monitored entities. Most work depends upon many sources of information, and has more interdependencies than can be represented by any one information source.

Other awareness tools attempt to obtain information from arbitrary information sources (e.g. they can monitor any type of document regardless of format). Such tools are possible because they are not required to understand the internal working of every information source, but can instead provide generalizations. For example, by monitoring a file system, one can become aware that documents are being changed, regardless of the document's format. However, while such awareness tools can provide information from a variety of sources, they cannot provide detail because they do not understand the internal changes made to the document.

Interchangeable Awareness Styles

Another problem with awareness is based on the fact that there is no optimal awareness style. An *Awareness style* is the manner in which notifications are sent and represented to the user of an awareness tool (e.g. sent to a desktop, wireless device, telephone; represented as text, sound or a variety of visualizations), and the manner in which the user is made aware of the arrival of a new notification.

For example, if one author is changing a document (e.g., a user's guide), a coauthor and an end user of the document are likely to benefit from different awareness styles. Specifically, a coauthor may want to be aware that changes are currently being made, and later that the document has been checked back into a version control system. On the other hand, end users of the document may not want to know until later in the day or week. They may prefer to receive the awareness information as a digest of all changes in the last time period, rather than being notified each time a change is made.

In other words, people need to be able to choose from a variety of awareness tools so that they can be made aware of information in a manner that best suits their needs and working styles. For this to be possible, awareness tools must be interchangeable; a person should be able to download an arbitrary awareness tool that supports an awareness style appropriate to that person's work practices, and use that tool instead of or in addition to any previous awareness tools.

Ubiquitous Awareness Infrastructures

To resolve these problems we have developed a software strategy we call CASS (Cross Application Subscription Service). We consider any type of software to be an information source. Whether the software is a user interface, a web server, a document or even a print driver, there are people who may be affected by changes to the state of the software, and who could perform their work more efficiently if they were aware of these changes. We also provide a notification server optimized for the distribution of awareness information. This server enables users to subscribe to types of information that they believe will affect them, and to specify to which types of awareness tools the information should be sent (enabling users to customize the awareness style).

Aspects of our goal are illustrated in Figures 1 and 2. In Figure 1, each person is the center of their own awareness network. They "tune" the strength of their awareness of various sources of information based on the extent to which they believe themselves

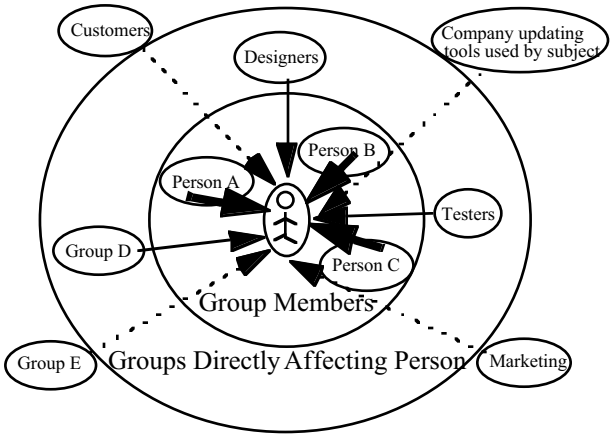


Figure 1: An awareness network. The strength of the awareness link is proportional to the entity's likelihood of affecting the person.

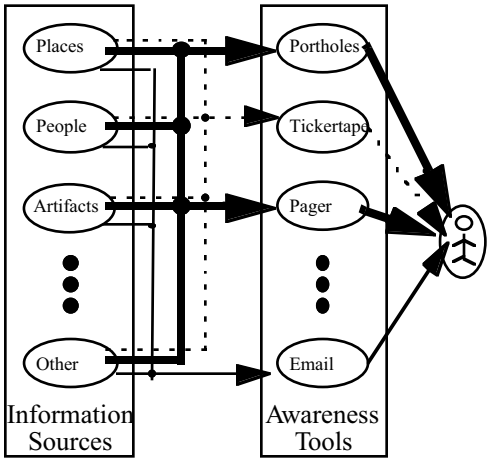


Figure 2: Strength of awareness is affected by style of awareness tool. Our goal is to enable any type of awareness tool to have some ability to represent any type of information source.

to be affected by the sources. Figure 2 illustrates two points. 1) as stated earlier, awareness tools need to be interchangeable; any good awareness tool must have some ability to represent awareness information from many sources. 2) users can tune the strength of their awareness through their selection of awareness tools. Awareness tools differ in the degree to which they are intrusive, and the choice of awareness tool directly affects the degree to which users will be aware of the tool's information.

The remainder of this paper presents background information, a scenario illustrating our goals, a detailed look at our approach, a demonstration of the feasibility of our approach through an implementation of the key aspects of the CASS strategy, and ends with a discussion of issues.

2 Background

Notification Server Architectures

On a technical level, our goal is to link arbitrary numbers and types of information sources, with arbitrary numbers and types of awareness tools, where both sources and tools are distributed across the Internet. Notification servers (Fitzpatrick, 1999, Ramduny et al., 1998, Carzaniga et al., 2000) are a common approach to providing such services. These technologies have focused upon enabling software to execute and interoperate over the Internet. Our goal; however, is to support human awareness rather than interoperation, and as such, the requirements for notification servers must change.

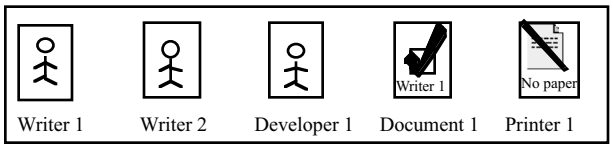
In the scenario of interoperation, software component X sends out an event that software component Y has subscribed to. Y subscribes to these events so that it can react quickly to the arrival of the event. If Y is not currently executing, then it will have no need to react to X. Humans, on the other hand, are not always on-line, and neither is their software. Relatively few notification services support a "polling" approach to notifications by which clients can request the events that occurred over the last time period. This is an essential capability when the goal is to support human awareness.

Furthermore, we are looking at asynchronous awareness, which has different speed requirements from synchronous awareness. Synchronous awareness tools such as shared white-boards (Streitz et al., 1994) enable users who are working on the same document to be aware of one another's actions as they happen. However, most human work is not synchronous in the sense that coworkers simultaneously working on different aspects of a task do not need to be notified of each pen stroke, key stroke, or decision the moment it happens.

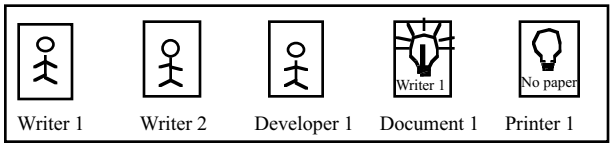
Awareness Tools

Researchers have provided a variety of tools for creating asynchronous awareness. Our goal here is not to replace them with better tools, but rather to begin to address the issue of providing a consistent way to supply these tools with their information. This will enable the tools to integrate more types of information, and to be easily replaced as future HCI research discovers better awareness techniques.

There are a variety of awareness styles demonstrated as effective within the HCI literature. Knowledge Depot (Kantor et al., 1999), for example, monitors work-related email and bboard discussions, and enabled users to maintain awareness of events and decisions made within these forums by emailing customized summaries of these discussions to interested parties. Portholes (Dourish, 1992) is an awareness tool which distributes photographs of offices once every five minutes to users. This enables users to be aware of whether coworkers are available by seeing whether they are on the phone, in a meeting, eating lunch in their office, out of the office, etc. Tickertape (Fitzpatrick, 1999), accomplishes a goal similar to that of CASS; it integrates detailed information from arbitrary information sources via the Elvin notification server and scrolls them across the top of a user's screen. Despite many similarities between our approach and Elvin/Tickertape, Elvin is optimized for interoperation rather than human awareness.



3a: Portholes has been instructed how to represent printer and document events by either users, developers or support staff.



3b:Generic icons can still be used to monitor arbitrary information sources when lacking instructions on how to represent events

Figure 3: A CASS Portholes window monitoring a variety of sources

3 Scenario

As a final step in motivating this work, we present a scenario illustrating different aspects of our goal. Consider a technical writer maintaining user documentation of a software component. This document is used by the customers who buy the component.

The technical writer uses the awareness tool sketched in Figure 3. Figure 3a represents a version of Portholes designed to be able to interpret arbitrary information sources. The first three images in the figure show the offices of coworkers that the technical writer works closely with. The fourth image shows an icon indicating that a monitored document has been checked out by Writer 1. The fifth image shows the status of the printer, which currently indicates that there is no paper. The icons displayed for the document and printer indicate an understanding of the information sources being monitored, but in fact a generic icon could also be used. Figure 3b shows the Portholes system in a slightly

altered scenario where the system has not been instructed (by users, developers or support staff) how to represent the user documentation being checked out or the printer being out of paper. A checked out document is instead represented by an active light bulb, and the inactive printer is represented by an inactive light bulb. This is explained in more detail in our implementation section.

It is important to note that the user could have subscribed to the same exact information using a CASS Tickertape tool. Using the version of the Portholes image grabber developed at NYNEX (Lee et al., 1997), an image can be represented as a single number representing the amount of change to take place within the office. This number indicates the presence or absence of people and activity. A tickertape might say:

Writer 1: *no change*... Writer 2: *some activity*… Writer 3: *some activity*... Document 1: *Writer 1*… Printer 1: *No paper*…

Other awareness tools would have access to the same information as Tickertape and Portholes, but each would have to represent it in its own style. Naturally some tools will be better at representing certain types of information, as an awareness tool that presents images can communicate far more detail from Portholes images than a textual or audio awareness tool, but it should be the user's choice whether that is detail that they need.

In this scenario, different awareness tools are used to maintain awareness of other information sources. Email is used to notify the technical writer of changes to meetings where he/she is a participant, and the Tickertape is used to display changes to meetings that involve the technical writer's boss (but not the writer). Knowledge Depot sends email summarizing discussions and decisions concerning changes to policies and other topics that can affect the writer.

Finally, the technical writer needs to be aware of changes made to functions of the software component that must be documented in the user guide. To do this, a developer within the company was able to easily put together a simple awareness tool (easy to implement because it used the services provided by CASS) which displayed a to-do list of both the relevant functions that have changed, and the names of the developer who made the changes. Using this to-do list, the writer knows who to contact for information on what must be changed in the documentation, and can check off items as the work progresses.

4 The CASS Strategy

The CASS strategy is a general specification of what must be done to achieve the goals discussed above. Our implementation of this strategy using CASSIUS (CASS Information Update Server) demonstrates a reasonable approach to implementing the CASS strategy.

Information from any source has the potential to affect the work of one or more people. We believe what is needed is an information environment consisting of workers' existing tools but where workers can subscribe to receive the information that they have determined can affect their work. Such an environment would need to provide the following services (numbers correspond to items in Figure 4):

1. Provide users with a mechanism to locate information sources affecting their work.
2. Isolate relevant subsets of information from that source.
3. Monitor sources and subsets for changes.
4. Allow people to subscribe to summaries of changes to the source or subset.
5. Send notifications of changes both to a notification server, and if needed, directly to the workers via email.
6. Represent notifications (or summaries of sets of notifications) within awareness tools.

To implement the strategy; 1) developers must build tools capable of monitoring state information and sending notifications of changes, 2) work environments must provide notification servers capable of informing awareness tools of both the types of events to which they can subscribe, and of new notifications that match the user's subscriptions, and 3) people must have awareness tools through which they can receive information from the notification server. Elements 2 and 3 have already been mentioned, so our presentation of the CASS Strategy will focus on element 1.

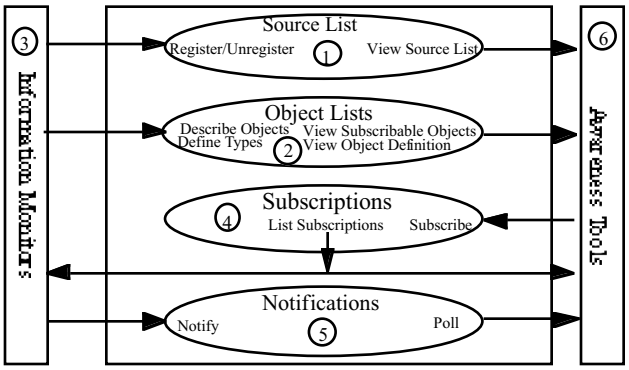


Figure 4: CASS Notification Server API

The Task of the Developer

Developers who want their tools to be able to send awareness information to people throughout a work environment must add three capabilities to their tools: Identify monitorable information, implement monitors, and add functionality for sending notifications.

1. Identify information that needs monitoring

Software developers will need to identify the information that can change within their systems and which may be of interest to users, their coworkers, and other involved parties. Objects worth monitoring may include sections of documents, parts of graphical objects, percentages of tests passed, numbers of hits on a web server, etc. More general packages (e.g. an office suite) may need to provide users with the ability to define for themselves what objects are monitorable. Thus, a user might be able to highlight a sentence of a document and have it be defined as a monitored object to which they are subscribed.

Identifying data to monitor is needed not just so that the tool can monitor the data, but also to inform users of the objects and properties to which they can subscribe. The notification server stores these objects in hierarchies which simplify the user's task of finding and subscribing to subsets of information sources.

2. Build monitors into software

There are a variety of approaches for monitoring changes, and again, the selection must depend upon the specific needs of the users. For instance, in the case of a tool for writing a user guide, developers may determine that the key objects to monitor are the sections and subsections of the document. Does the tool detect whenever a person is working on that section and send out an event? Does it wait for the user to save, and then generate an event based on a comparison between the current version and the previous version? Or, does it assume that many of the save operations will be to make sure work is not lost in case of a crash, rather than because the changes are finished? In this case, monitoring might instead happen when the author closes the document or issues a publish command after all changes for a version have been completed.

3. Provide functionality to send notifications

Finally, notifications of changes must be sent to the users. The CASS strategy recommends using two approaches to this: notification server oriented and application oriented.

The notification server approach has all notifications sent to a notification server, which combines many notifications and sends them to the awareness tools that the user has selected. There are three key benefits to this approach: 1) people can select the desired types of awareness tools, specifying which information should go to pagers, email, portholes, etc., 2) the server can be used to browse through lists of information sources to which they can subscribe, 3) users do not need to be able to access the information source directly.

The application-oriented approach does not assume the presence of a notification server within an environment, and ensures that the lack of such a server will not prevent users from becoming aware of information. Users who have direct access to an information source (e.g., the authors of a document) should be able to subscribe to notifications of changes directly within the application. The application sends notifications directly to the user as email messages.

The benefits of the application-oriented approach are 1) it does not depend upon a notification server being available — the email server is the notification server, 2) users can view the objects while deciding to what they should subscribe — the notification server can only describe the objects. 3) security is improved — many people can access a server, but only a select few can access the version of a document that authors are writing.

The CASS strategy recommends providing both approaches and letting users decide how to use it based on their work environment (except for tools that do not support user interfaces, such as device drivers).

5 Implementing CASS Strategy

Our strategy requires three types of software: notification server, information sources, and awareness tools. We have implemented a small sample of each of these software types to validate our approach and demonstrate its feasibility.

Notification Server

We have implemented a set of services that we believe best addresses the goals of the CASS strategy. Our CASS Information Update Server (CASSIUS) provides the services shown in the center box of Figure 4. Figure 5 shows the important elements of a notification sent from an information source to CASSIUS. To provide a polling approach to access, as well as scalability, and ease of access from a wide variety of types of software, all of the services described here are

accessed using HTTP (Hypertext Transfer Protocol). The following sections describe key services of the CASSIUS notification server.

Source List

Information sources must register, supplying names, passwords and descriptions (#1 of Figure 4). Users can scan through lists of account names and descriptions looking for relevant sources to which they can subscribe. Our implementation also provides mechanisms for categorizing information source types (e.g., design documents, testing programs, research papers, documents from a specific author). This categorization results in the use of "AccountPath" in Figure 5 to identify sources using a name and type pair.

Furthermore, as our view is that all software should monitor its state and send out awareness information, the act of registering and unregistering new information sources causes CASSIUS to send a notification to itself. This allows people to subscribe and maintain an awareness of the availability of new information sources, and to even specify that they only want to be aware of new information sources of a particular type.

Object Lists

Information sources send (and maintain) hierarchical lists of objects to the notification server (using service #2 of Figure 4). Users navigate those hierarchies, finding the information to which they want to subscribe. CASSIUS enables information sources to optionally associate type-names with objects, and to define what kinds of events those types can receive. This enables users and awareness tools to focus on specific types of events to affect the object of interest. However, use of this feature is optional, as there is also a generic set of event types that must be used in addition to any custom event types. Generic event types enable notifications to be interpreted by many awareness tools (see Figure 3b).

Subscriptions

A service for listing subscriptions enables the awareness tool to see what it has previously subscribed to, and allows information sources with many objects that constantly change to determine in which objects users have an interest (#4 of Figure 4). This prevents it from needing to send notifications for every change. Simpler applications may decide to send out notifications for every event, letting the notification server dispose of those to which no one has subscribed.

Notifications

The notification, whose key fields are illustrated in Figure 5, focuses upon information needed for awareness. The most common field for creating awareness is the one-line summary, used in email subject lines, tickertape summaries, and other types of textual and verbal awareness tools. Such summaries are generally linked to the contents or URL of the "Data" field of Figure 5. The "Data" field is also used for binary data used in non-textual awareness tools.

Summary	One line textual summary of the change
GenericEvent	An event name chosen from a list of generic event names understood by all CASS applications and awareness tools
Event	Information sources define object types and list possible events for each type. This field is an event name from the appropriate event list.
Value	Optional numerical value to represent the change
Data	Optional link or a MIME attached document describing the details of the change.
Person/Place	Optional person or place associated with event.
ObjectID	Identifies the object or property that has changed
AccountPath	Identifies the information source

Figure 5: Major elements of a CASSIUS notification

The person/place field enables subscribers and awareness tools to focus on events generated by people or places they want to be aware of. A CASS Portholes tool, should it lack a snapshot of an office, could easily replace the photo of the office with other information that comes from the same office, enabling alternate ways of representing activity within the location.

Information Sources

The second phase of implementing the CASS strategy was to implement and instrument information sources (#3 of Figure 4). Our first application was a simple word processor. Users select sections within the document and subscribe to be emailed notifications of changes to those sections. The editor also sends all notifications to CASSIUS and maintains the object hierarchy within CASSIUS as users create, delete, and move sections.

We also added a component to Knowledge Depot (Kantor et al., 1999) for monitoring activity within the server, and sending events whenever the server goes on or off-line. Notifications are also sent when new messages are posted, enabling people to use a variety of awareness tools to monitor group discussions.

Finally, we instrumented the WebDav Apache plug-in (mod_dav) to send notifications to CASSIUS whenever a file is checked in or out, locked or unlocked, created or deleted. Creating, deleting and moving files and folders also causes equivalent changes to be made to the object hierarchy on CASSIUS. This lets users find files in the hierarchy the same way they find them in WebDav's file hierarchy.

Awareness Tool

For the third aspect of implementing the CASS strategy (#6 of Figure 4), we built the CASSandra toolkit for creating awareness tools. It provides functionality for enabling users to see lists of information sources, and to subscribe to both objects within those information sources and events affecting those objects. Developers link in their own user interface for displaying notifications, where the user interface determines the awareness style. This toolkit was used to build a variety of awareness tools/styles including a tickertape awareness tool.

6 Discussion

Software Development Effort

The above systems were built as a proof of concept, but the actual effort to build these systems and to instrument existing systems to monitor their own state was minimal. Instrumenting Knowledge Depot and WebDav to send notifications was a matter of a couple of hours of work. Building and testing CASSIUS took only a couple of weeks, and the parts of the CASSandra awareness toolkit that polls for new data and parses the server responses took under an hour.

Naturally, the actual difficulty will vary with both the nature of the information source's data-structures and the needs of the users. However, the value of the information to users and their coworkers, and the value of receiving the information in a style that best fits each user's work practices makes the benefits of this additional effort well worth while.

The service itself proved to be very simple to use, being accessible with basic commands available in many scripting and compiled programming languages (e.g. any language that can send HTTP requests and parse textual responses can use CASSIUS).

Implementation and Validation

As discussed above, we were able to implement all of the components of our CASS strategy, thus demonstrating that our approach is, to some degree, feasible. With this starting point, let us now consider how well the CASS strategy addresses our goals of supporting interchangeable awareness tools and resolving the detail-variety trade-off.

Creating Interchangeable Awareness Tools

We addressed the problem of making awareness tools interchangeable by providing a consistent notification format that can be interpreted by any well designed awareness tool, and by allowing the awareness tool to subscribe to any accessible information source. While much of the proof of our ability to achieve this goal has been through scenarios (Figure 3), we have also demonstrated that such tools can in fact be built using the CASS strategy.

Resolving the Detail-Variety Trade-off

We insured the availability of variety by providing a set of services to enable *any* software to send awareness information. We insured the availability of detail by making it the responsibility of the software to understand its own data-structures and to report on changes. It took us relatively little effort to instrument our information sources, indicating that this is a reasonable approach to resolving the trade-off.

Facilitating Adoption

While the services were easy to provide and to use, the tasks ahead are far more challenging. We will need to 1) Build up a diverse information environment, finding more systems such as WebDav to instrument and work with. With this done, users can maintain awareness of a diverse set of information from their work environment, and we can study the collaborative effect of providing the environment. 2) Build up a local user community of people who use the awareness tools and who will CASS-enable applications that they build. 3) Attempt to gain wider adoption of our approach. 4) Address concerns described in the remainder of this paper.

Information Overload

Information overload is always a problem (Maes, 1994). We have tried to address this issue by enabling users to control what information they receive and choosing the tools to represent that information. Tools that summarize and visualize information sets for the user can reduce overload.

Tickertape applications also provide an interesting technique for avoiding information overload for low priority information. Unlike email, where the model is for a person to give some attention to everything that arrives in a mailbox (even if it is only enough attention to give a delete command), a tickertape's attention demands are a constant with respect to the quantity of notifications a user has received. This is made possible by the fact that it is designed to have users glance occasionally at it, and obtain a random selection of notifications.

There is no assumption in this type of tool that a user will see all notifications (which is why this is appropriate only for lower-priority information), and as the quantity of notifications increase, the user will observe a lower percent of those notifications.

However, if there are a large number of high-priority notifications that users have configured to receive either by email or by other tools that are less able to handle information overload, the problem will remain.

Efficiency

We discussed before that we were interested in asynchronous collaboration, and could ignore the needs of synchronous tasks, which would have required the server be able to push information to awareness tools as soon as the information arrives. Unfortunately, the border between synchronous and asynchronous is not solid, and there may be tasks we will want to support in the future that require greater efficiency than polling the server can provide. Future versions of this work may need to account for that possibility, perhaps by supporting both client polling and server push.

7 Conclusions

The CASS strategy resolves the detail-variety trade-off and the lack of interchangeability of awareness tools. By removing these limitations of awareness tools, we have enhanced the ability for software to support awareness environments, where each user is at the center of his or her own awareness network. We have also provided an implementation of this strategy, and demonstrated it through the CASSIUS notification server.

The implication of this work on HCI research are that 1) HCI researchers can focus on developing new awareness styles, without having to also develop the infrastructure and information sources to supply the tools with information, and 2) tools which need to make users aware of state can use the style chosen by users and implemented by awareness tool developers, rather than requiring system builders to implement highly configurable interaction techniques within their systems.

Acknowledgements

Current research support for this work is provided by the National Science Foundation under grants CCR-9624846 and CCR-0083099, by UC Irvine.

References

Carzaniga, A., Rosenblum, D. and Wolf, A. (2000) Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service In *Principles of Distributed Computing,* Portland OR

Dourish, P., Bly, S. (1992) *Portholes: Supporting Awareness in a Distributed Work Group* In *CHI'92* ACM, pp. 541-547.

Fitzpatrick, G., Mansfield, T., Arnold, D., Phelps, T., Segall B., Kaplan, S. (1999) *Instrumenting and Augmenting the Workaday World with a Generic Notification Service called Elvin* In *ECSCW'99* Copenhagen.

Gaver, W. W., R. B. Smith, and T. O'Shea (1991) Effective Sounds in Complex Systems: The ARKola Simulation In *CHI'91*, New Orleans.

Gutwin, C., Greenberg, S. (1998) Effects of Awareness Support on Groupware Usability In *CHI'98* ACM, Los Angeles, CA, pp. 511-518.

Kantor, M., Redmiles, D. and Zimmermann, B. (1999) Supporting Awareness and Coordination Between Groups, UC Irvine, Report #UCI-ICS-99-46

Lee, A., Girgensohn, A. and Schlueter, K. (1997) NYNEX Portholes: initial user reactions and redesign implications In *CHI'97*, pp. 385-394.

Maes, P. (1994) Agents that Reduce Work and Information Overload in *CACM,* **37,** 31-41.

Ramduny, D., Dix, A. and Rodden, T. (1998) Exploring the design space for notification servers In *CSCW'98* ACM, Seattle, pp. 227-235.

Sibert, L. and Jacob, R. (2000) Evaluation of Eye Gaze Interaction In *CHI'2000* ACM, The Hague, The Netherlands, pp. 281-288.

Streitz, N. A., Geiler, J., Haake, J. M. and Hol, J. (1994) DOLPHIN: integrated meeting support across local and remote desktop environments and LiveBoards In *CSCW'94* ACM, pp. 345-358.

Williamson, C. and Shneiderman, B. (1992) The dynamic HomeFinder: evaluating dynamic queries in a real-estate information exploration system In *Research and development in information retrieval* ACM, Copenhagen, pp. 338-346.